On-line algorithms for expert advice and task systems (thesis proposal)

Carl Burch

February 16, 1999

Abstract

This thesis explores two general on-line scenarios historically deriving from two largely disjoint communities. Though the problems are inherently similar, the techniques and questions developed for these two problems are very different; an examination of them provides new insights and new directions, each enriching the other. From machine learning comes the problem of *predicting from expert advice*—that is, of choosing one of several experts for each query in a sequence without doing much worse than the best expert overall. And from competitive analysis comes *metrical task systems*, where the algorithm is to decide in which state to process each of several sequential tasks, where each task specifies the processing cost in each state, but changing states also has a cost according to a metric.

Besides connecting these problems, the thesis presents a variety of results lying in and around the area defined by them, including interesting hybrid problems (adding a switching cost between experts) or generalizations (a task system where the algorithm occupies several states). This thesis answers a long-standing question in competitive analysis by proving the first polylogarithmic competitive ratio for task systems in general metric spaces.

1 Motivation

Selecting an action without knowing the future is an essential problem in many scenarios, including caching, behavior prediction, and stock investment. In such scenarios one attempts to select an action that is nearly as good as the best possible action, predicting (in a sense) which is best for the future.

As a motivating problem, consider process migration. We have a process moving between computers on a network. For each five-minute period, the process chooses a computer on which to run and then receives all computers' load averages for that period. How should the process decide where to go each period?

Researchers have naturally considered such problems in the past, but traditionally they assume simple distributional assumptions about how the world behaves. Such a model may not be accurate. For example, assuming independent draws from a distribution is tempting, but it doesn't model our migration problem very well: A machine busy in the last period will probably also be busy this period. Of course one can generalize the probabilistic assumption to incorporate state, but a more radical solution is to ask how well one can do without any probabilistic assumptions at all. How, indeed, should we decide on migration if the network is adversarial? Being able to predict in such an environment may seem impossible, but there are algorithms that do surprisingly well.

A crucial feature of many of these algorithms is their employment of randomization. The envisioned scenario is that the adversary is *oblivious*—that is, it knows our strategy but does not see our specific random choices. This weak assumption, the only one made on the adversary's nature, allows much better theoretical bounds.

Machine learning theory and competitive analysis are two areas that have studied such distribution-free problems. This thesis studies two fundamental problems in these areas: prediction from expert advice and metrical task systems [LW94, BLS92]. The goal of the thesis is to explore the relationship between these two problems, with the specific aim of exploiting past ideas to achieve improved results for these problems and their variants.

By relating and studying prediction from expert advice and metrical task systems, this thesis advances the basic understanding of on-line selection and prediction in adversarial environments. This study includes an examination of algorithms for several natural and related on-line problems, and it describes several new algorithms with provably good behavior. The underlying goal of this work is to develop a unified picture of what one can do in adversarial on-line selection scenarios.

2 Background

To understand the thesis results in detail, we must understand these two problems and their background. This section is meant to introduce the reader to the problems; the following sections describe the specific results appearing in the proposed thesis.

2.1 Prediction from expert advice

The problem of prediction from expert advice has many specific formulations. The "decision-theoretic" formulation (which we term *Experts-DTF*), proposed by Freund and Schapire, is most useful for our purposes [FS97].

2.1.1 Problem statement

Say there are n experts who make predictions at each time step. (These "experts" may be stock market analysts, for example.) As the on-line algorithm, we are to choose an expert for each time step. After we make our choice, we learn how badly the experts did with a *loss vector*, specifying the badness of each expert's prediction, in the range $[0, 1]$. Our cost is the loss for the expert we chose, and we wish to bound the sum of losses over time.

Prediction from expert advice is a fundamental problem in the theory of on-line machine learning. It is meant to model a situation where we have a set of hypotheses (termed *experts* in the problem formulation), one of which predicts fairly well how the world operates. The question is how quickly we can converge on a good predictor. Thus, our goal is to bound how much worse we do relative to the best single expert.

In contrast to much of machine learning, the expert advice problem does not employ distributional assumptions. That is, the experts need not perform uniformly over time in any sense. Despite the absence of such assumptions, the theoretical bounds obtained are surprisingly good.

2.1.2 Noiseless data

A particularly simple special case is when one of the hypotheses is perfect (that is, never incurs any loss). It is not too hard to see that in this case a *deterministic* algorithm must incur a worst-case loss of at least $n - 1$: An adversary could easily construct a loss sequence by simulating the algorithm and for each time step giving a loss of 1 to the expert which the algorithm chooses, until only one expert is left. The bound of $n - 1$ is quite undesirable; we can do much better employing randomization.

The following algorithm is Rand-Halving, a randomized version of the well-known Halving algorithm (a deterministic algorithm that applies only when the algorithm can see the experts' predictions and the predictions are binary-valued; see for example Mitchell [Mit82]) and a degenerate instance of the Hedge algorithm discussed below [LW94, FS97].

Algorithm Rand-Halving**:** *Pick an expert uniformly at random. When it errs, we pick a new expert uniformly at random from among those that have performed perfectly so far. We continue repeating this.*

We analyze the performance of this very simple algorithm to give an idea of the type of analysis involved.[1]

Theorem 1 *If one of the experts is perfect (ie, never incurs any loss), then Rand-Halving has expected loss at most $H_n - 1$. (Here H_n represents the nth harmonic number, $\sum_{i=1}^{n} 1/i \approx \ln n$.)*

Proof. Let L_t be a random variable denoting the amount of loss our algorithm incurs in round t; Rand-Halving has expected total loss $\mathbf{E}\left[\sum_t L_t\right] = \sum_t \mathbf{E}\left[L_t\right]$. Also, let k_t represent the number of experts erring in the tth time step that have not erred before; by assumption, $\sum_t k_t \leq n - 1$.

In the first round, we choose a random expert and so have a k_1/n chance of incurring a loss; if we do, the problem formulation restricts the loss to at most 1. Thus the expected first-round loss $\mathbf{E}\left[L_1\right]$ is at most k_1/n. Our second-round choice is a uniform-random one of the $n - k_1$ perfect experts left, so we have $\mathbf{E}\left[L_2\right] \leq k_2/(n - k_1)$. In general, on the tth time step we are at a uniform-random expert among the $n - \sum_{t'<t} k_{t'}$ remaining experts and so have $\mathbf{E}\left[L_t\right] \leq k_t/(n - \sum_{t'<t} k_{t'})$. Using these facts, we can now bound our total loss.

$$
\begin{aligned}
\sum_t \mathbf{E}\left[L_t\right] &\leq \frac{k_1}{n} + \frac{k_2}{n - k_1} + \frac{k_3}{n - (k_1 + k_2)} + \cdots \\
&= \sum_{i=0}^{k_1 - 1} \frac{1}{n} + \sum_{i=0}^{k_2 - 1} \frac{1}{n - k_1} + \sum_{i=0}^{k_3 - 1} \frac{1}{n - (k_1 + k_2)} + \cdots \\
&\leq \sum_{i=0}^{k_1 - 1} \frac{1}{n - i} + \sum_{i=0}^{k_2 - 1} \frac{1}{n - k_1 - i} + \sum_{i=0}^{k_3 - 1} \frac{1}{n - (k_1 + k_2) - i} + \cdots \\
&\leq \sum_{i=0}^{n-2} \frac{1}{n - i} \\
&= H_n - 1
\end{aligned}
$$

■

2.1.3 Noisy data

When all experts incur some loss, the problem becomes more complicated. It turns out that a very simple algorithm does well in this case. The Hedge algorithm is Freund and Schapire's DTF-Experts adaptation of Littlestone and Warmuth's weighted majority algorithm [LW94, FS97].

Algorithm Hedge**:** *We employ a parameter $\beta \in (0, 1)$ and maintain a weight w_i for each expert, initially 1. At each time step we choose an expert randomly, choosing expert i in proportion to its weight (that is, probability $w_i / \sum_{j=1}^{n} w_j$). After receiving loss vector ℓ, we update each expert i's weight to become $w_i \beta^{\ell_i}$.*

[1] Our definition of Rand-Halving is peculiar for machine learning algorithms because it attempts to remain at an expert until it errs. An alternative is to independently select a hypothesis each time step uniformly from among those who have not erred. This alternative has the advantage that it works against an adaptive adversary (the stated algorithm assumes an adversary oblivious to the algorithm's past random choices) but the twin disadvantages that it switches hypotheses more often than necessary and that it obscures the parallels with task systems drawn below.

The following fundamental theorem, which we do not prove here, bounds the total loss of Hedge relative to the best expert.

Theorem 2 ([FS97]) *If the best expert incurs a total loss of m, then Hedge incurs expected loss at most*

$$\mathbf{E}\left[loss_{\textsf{Hedge}}\right] \leq \frac{\ln 1/\beta}{1-\beta} m + \frac{1}{1-\beta} \ln n \; .$$

Remark. To make better sense of this bound, let $\beta = 1/(1+\varepsilon)$. Then the above translates to approximately $(1+\varepsilon/2)m + (1+1/\varepsilon)\ln n$. Intuitively, this is an explicit trade-off between how quickly we settle on a particular expert (the $(1+1/\varepsilon)\ln n$ term) and how quickly we are able to adapt if that expert is actually bad but happens to do well for the first several rounds (the $(1+\varepsilon)m$ term).

This bound is very close to the best expert's loss. Moreover, it says that we can double the number of experts (refining the hypothesis space by a factor of two) with very little increase in worst-case loss. A major strength of the theory of expert advice is how tight a bound we get with the very simple algorithm Hedge. (In fact, the coefficients of Theorem 2's guarantee are optimal for on-line algorithms [Vov95, FS97].)

2.1.4 Partitioning bound

The above algorithms are meant to work well with respect to the best single expert over all time steps. The *partitioning bound* is a more ambitious goal. Here we try to do well against any partitioning of time into intervals, where we pick the best expert within each time interval of the partition. Being able to do well against all partitions includes scenarios where one expert does very well for the first half of time, whereas another expert does best on the last half of time. Algorithms to do well with this bound must be able to quickly adapt to changed expert performance.

Both the WML algorithm of Littlestone and Warmuth and the Fixed-Share algorithm of Herbster and Warmuth are variations of the weighted majority algorithm; they work to achieve this type of goal [LW94, HW98]. (Blum and Burch adapt these algorithm to the DTF-Experts setting and respectively call them Thresh and Share [BB97].) Given a partition P, let k_P denote the number of intervals and L_P denote the sum over intervals of the loss incurred by the best expert within that interval. Both algorithms have the property that their total expected loss is bounded by (approximately)

$$\mathbf{E}\left[loss_A\right] \leq (1+\varepsilon)L_P + \left(1+\frac{1}{\varepsilon}\right) k_P \ln n \tag{1}$$

for any partition P.

2.2 Metrical task systems

The *metrical task system* (MTS) problem, proposed by Borodin, Linial, and Saks, is fundamental to competitive analysis in the same sense that the expert-prediction problem is fundamental to on-line learning theory: It abstracts many important on-line problems [BLS92].

2.2.1 Problem statement

Say we are controlling a system of n *states* with a distance metric d between them. At all times our on-line algorithm occupies a state. Each time step, we receive a *task vector*, specifying a cost in each state (representing the cost to process the task in each state). When we receive a task vector t, we choose a state in which to process it. We pay both for moving between states (according to d) and for processing the task (according to t). Our goal is to minimize our total cost over the task sequence.

In the task system problem, as with all of competitive analysis, the accepted goal is to demonstrate an algorithm achieving a small *competitive ratio*. Let $cost_A(\sigma)$ be the cost incurred by algorithm A on a sequence σ of tasks. (This is a random variable if A is randomized.) Similarly, let $cost_{\mathrm{OPT}}(\sigma)$ be the optimal cost achievable on σ—that is, the least cost we can achieve given the task sequence in advance (with no bounds on computation time). An on-line algorithm A is said to have competitive ratio ρ if for each task sequence σ, its cost obeys the inequality

$$\mathbf{E}\left[cost_A(\sigma)\right] \leq \rho \cdot cost_{\mathrm{OPT}}(\sigma) + \zeta\ ,$$

where ζ is a number independent of σ (but may depend on n or d). (This definition assumes an *oblivious adversary model*; an adversary must choose the sequence before knowing A's random choices. This model is important to our analysis, since an algorithm competing against an adversary who can see A's random choices cannot do better than a deterministic algorithm [BDBK+94].)

The importance of metrical task systems lies in the fact that they generalize many natural on-line problems. The paging problem, for example, can be formulated in the metrical task system framework. If we have a cache that can hold k items, and there are n items in the universe, then each state of the task system would be one of the $\binom{n}{k}$ choices of k items from the n. Our current state tells us what we should hold in our cache. A request to page i is represented as a task which has a cost of 0 for those states where i is in the state's corresponding cache and ∞ elsewhere. The metric between states is the number of page loads required to move between the two sets.

2.2.2 Uniform metric

One of the most important and well-understood MTS cases is the case of a uniform metric, where $d(u, v) = 1$ when $u \neq v$. The Mark algorithm of Borodin, Linial, and Saks has a competitive ratio of $2H_n$ [BLS92]. (This algorithm is similar to the paging algorithm of the same name by Fiat *et al* [FKL+91].)

Algorithm Mark: *We proceed in phases. At the beginning of the phase all states are unmarked, and we choose a random state to occupy. As tasks are received, we increase counters on each state, keeping track of the total processing cost for that state in this phase. When a state's counter reaches 1, we say that this state is* marked. *When our current state becomes marked, we move to a random unmarked state. When all states are marked, we reset all marks and counters and begin a new phase.*

The appearance of H_n in the performance of both Rand-Halving and Mark is not coincidental. Looking at the problems and algorithms, we see that Rand-Halving is basically Mark (where "unmarked" means that the expert has not yet erred), except that in the noiseless Experts-DTF problem, the first phase never comes to an end. Because it is very forgetful (losing all data each phase), Mark's behavior is very different from Hedge, losing as much as $O(\log n)$ times the best single expert's loss.

On the hardness side, every on-line algorithm for the uniform metric has a competitive ratio of at least H_n [FKL+91, BLS92]. (Irani and Seiden give an algorithm achieving $H_n + O(\sqrt{\log n})$ [IS98].)

2.2.3 General metrics

Besides introducing the problem and presenting Mark, Borodin, Linial, and Saks also demonstrate an algorithm for general metric spaces with competitive ratio $2n - 1$. Although the definition of competitive ratio allows for randomized algorithms, theirs is a deterministic algorithm, and they show that $2n - 1$ is optimal for deterministic algorithms. Subsequent work by Irani and Seiden demonstrates a randomized algorithm beating this (ratio $1.58n - 0.58$) [IS98].

The example of the uniform metric leads to the conjecture that a general algorithm exists achieving $O(\log n)$ on every metric. Although such an algorithm remains elusive, an algorithm by Bartal, Blum, Burch, and Tomkins achieves ratio $O(\log^5 n)$ [BBBT97]. More discussion of this result appears in Section 3.3.

2.2.4 Unfair competitiveness

Critics of competitive analysis observe that the competitive ratio has many problems. In many important problems, the proven bounds are quite high. (Neither the deterministic $2n - 1$ bound nor the randomized H_n bound is very inspiring.) Also, the competitive ratio sometimes indicates that algorithms should do well when in practice they perform poorly. (For paging algorithms, FIFO has the same theoretical guarantee as LRU, although LRU is far superior in practice [ST95].)

The competitive ratio has a single parameter describing the "hardness" of an event sequence. One alternative approach developed in the thesis is to refine the bound for problems involving more than one cost, like metrical task systems' task-processing costs and movement costs. This is the notion of *unfairness* arising from metrical task system work by Blum *et al* and Seiden and crucial to the general-metric $O(\log^5 n)$ bound [BKRS92, Sei98, BBBT97]. Intuitively, the algorithm competes against an adversary who pays just as much as the on-line algorithm in one cost but is handicapped by an r factor in another cost. Say $cost_{task}(\sigma, \tau)$ is the task-processing cost on task sequence σ for action sequence τ, and say $cost_{move}(\sigma, \tau)$ is the distance-metric cost. Then we say on-line algorithm A has *r-unfair competitive ratio ρ* if, for all input sequences σ, algorithm A (with r as a parameter) uses an action sequence $\tau_{A,\sigma}$ so that for all action sequences τ,

$$\mathbf{E}\left[cost_{task}(\sigma, \tau_{A,\sigma}) + cost_{move}(\sigma, \tau_{A,\sigma})\right] \leq \rho(cost_{task}(\sigma, \tau) + r \cdot cost_{move}(\sigma, \tau)) + \zeta\ .$$

(Again, ζ is an additive constant that is independent of the input sequence σ but may depend on the specific problem.)

Notice that this notion of unfairness bears a strong resemblance to the Experts-DTF partitioning bound. In particular, choose $cost_{task}(\sigma, \tau)$ to be the loss incurred by prediction sequence τ and $cost_{move}(\sigma, \tau)$ to be the number of times τ switches experts. If we have an algorithm A with r-unfair competitive ratio $1 + (\ln n)/r$ for the Experts-DTF problem, then we have an algorithm achieving partitioning bound (1): Choose $\varepsilon = (\ln n)/r$; any partition P corresponds to an action sequence τ_P with $cost_{task}(\sigma, \tau_P) = L_P$ and $cost_{move}(\sigma, \tau_P) = k_P$.

$$\begin{aligned}
\mathbf{E}\left[loss_A\right] &\leq \mathbf{E}\left[cost_{task}(\sigma, \tau_{A,\sigma}) + cost_{move}(\sigma, \tau_{A,\sigma})\right] \\
&\leq \left(1 + \frac{\ln n}{r}\right)(cost_{task}(\sigma, \tau) + r \cdot cost_{move}(\sigma, \tau)) + \zeta \\
&= (1+\varepsilon)L_P + \left(1 + \frac{1}{\varepsilon}\right) k_P \ln n + \zeta\ .
\end{aligned}$$

Although such a competitive ratio for Experts-DTF may appear overambitious, Bartal *et al* demonstrate a similar bound for the closely-related uniform-space MTS problem [BBBT97].

3 Completed results

With a reasonable understanding of the state of expert advice and task system research, we are ready in this section to examine the specific expected contributions of the thesis. The following subsections delineate specific results that will appear in the thesis.

3.1 Comparison of problems and algorithms

The experts and loss vectors in the Experts-DTF setting appear nearly identical to the MTS problem's states and cost vectors. But the problems also have several differences.

- The Experts-DTF goal of doing well relative to the best expert is very different from the goal of doing well relative to the best off-line strategy. (But unfair competitiveness, which lies between the two, is natural in both contexts.)
- The MTS problem has a cost for switching between states, a concept absent in the Experts-DTF problem.
- An MTS algorithm has *one-step lookahead* (the algorithm chooses the state after it sees the task vector) whereas an Experts-DTF algorithm has *zero lookahead* (choosing the expert before seeing the loss vector).
- To make zero-lookahead tractable, the Experts-DTF algorithm bounds the loss per time step by 1.

As Blum and Burch observe, these differences are not always substantial. Indeed, any uniform-space MTS algorithm with a reasonable competitive ratio translates to an Experts-DTF algorithm with a partitioning bound (though not necessarily as good as bound (1)).

Theorem 3 ([BB97]) *Let A be a randomized uniform-space MTS algorithm achieving r-unfair competitive ratio $\rho_{n,r}$. Then it can be used in the oblivious Experts-DTF setting with a partitioning bound of*

$$\mathbf{E}\left[loss_A\right] \leq \rho_{n,r} \cdot L_P + \rho_{n,r} r \cdot k_P + \zeta \ ,$$

for any k_P-segment partition of loss L_P. (Here ζ is the additive part to the competitive ratio; typically we have $\zeta \leq r$.)

Proof. Consider a sequence of trials σ. Let $loss_A$ denote the loss of algorithm A on σ and $cost_A$ denote the expected cost of A viewed as tasks in a uniform MTS problem. We know $\mathbf{E}\left[loss_A\right] \leq \mathbf{E}\left[cost_A\right]$ since, for each trial σ_t, if $\sigma_{t,i}$ represents the loss of expert i in the trial, and $p_{t,i}$ represent the probability A is at expert i before receiving loss vector σ_t then we can bound the Experts-DTF loss as

$$\begin{aligned}
\mathbf{E}\left[loss_{A,t}\right] &= \sum_i p_{t,i}\sigma_{t,i} \\
&= \sum_i p_{t+1,i}\sigma_{t,i} + \sum_i (p_{t,i} - p_{t+1,i})\sigma_{t,i} \\
&\leq \sum_i p_{t+1,i}\sigma_{t,i} + \sum_{i: p_{t,i} > p_{t+1,i}} (p_{t,i} - p_{t+1,i})\sigma_{t,i} \\
&\leq \sum_i p_{t+1,i}\sigma_{t,i} + \sum_{i: p_{t,i} > p_{t+1,i}} (p_{t,i} - p_{t+1,i}) \\
&\leq \mathbf{E}\left[cost_{A,t}\right] \ .
\end{aligned}$$

The second inequality represents the fact that each expert has loss at most 1 per trial; the final step represents the fact that in trial t, as an MTS algorithm, A would pay the first term in task-processing cost and at least the second term in movement cost.

Consider a partition that incurs loss L_P and switches between experts k_P times. This is equivalent to an MTS action sequence paying L_P for task processing and k_P for movement. Because A has unfair competitive ratio $\rho_{n,r}$ we know

$$cost_A \leq \rho_{n,r}(L_P + rk_P) + \zeta \ ,$$

for some ζ, as desired. ■

The historically-developed algorithms for these two problems are quite different, but through analyzing them a picture emerges showing some interesting relationships. (The following results appear in Blum and Burch [BB97].)

- Littlestone and Warmuth develop Thresh, which does well in the Experts-DTF setting [LW94]. But Thresh does not control its movement very well and so has an unbounded competitive ratio.
- Herbster and Warmuth develop Share, which performs comparably to Thresh in the Experts-DTF setting [HW98]. Unlike Thresh, however, it performs well in the MTS setting also, achieving a reasonable unfair competitive ratio.
- The uniform-space MTS algorithm Mark has r-unfair competitive ratio $(1 + 1/r)\log n$ [BLS92]. Theorem 3 immediately implies a partitioning bound, but one much worse than the bound (1) achieved by Share and Thresh.
- The uniform-space MTS algorithm Odd-Exponent developed by Bartal *et al* for their general-metric MTS result has r-unfair competitive ratio $1 + (2e\ln n)/r$ [BBBT97]. Applying Theorem 3 (with $r = (2e\ln n)/\varepsilon$) gives bounds similar to the bound achieved by Share and Thresh.

3.2 Experts-DTF with state

Another direction to go in exploring the two problems is to hybridize them. Consider the Experts-DTF setting where we add a cost d for changing experts. That is, the algorithm may sit on an expert for several rounds, but when it changes it must incur an automatic additional loss of d. This problem has not been considered previously, but it is a natural one. For example, we might want to use an Experts-DTF algorithm to decide where to save our money. Unfortunately, the Experts-DTF model ignores the issue of account-opening charges. The addition of a switching cost models this additional aspect.

This modified version of Experts-DTF has implications in competitive analysis, since with it we can model situations where we want to do as well as the best of several on-line algorithms. A particularly notable instance of this is in paging, where we might have several paging algorithms running (maybe FIFO, MFU, and LRU). We can simulate all of these on-line and use Hedge to select between them, but changing our algorithm may require us to change caches, at a cost of at most k (if the cache holds k pages).

In the noiseless case, as we can see in the proof of Theorem 1, Rand-Halving expects to move at most $H_n - 1$ times, so we can say that if one expert is perfect we achieve an expected total loss of at most $(1 + d)(H_n - 1)$, which is optimal for the noiseless case.

If all the experts incur some loss, then since the multiplicative-update algorithm does not attempt to control the number of times we switch experts, it is not immediately obvious how to approach the problem. But some analysis shows that, using a conservative switching strategy, Hedge also achieves a reasonable bound in this scenario.[2]

[2] By *conservative switching*, we mean the following: Maintain probabilities $p_{t,i}$ for selecting expert i for time step t. Of course at any time step t we are actually at a particular expert i, and we must maintain the probabilities while deciding to move only minimally. We do this as follows. If $p_{t,i} \le p_{t+1,i}$, then we do not move. Otherwise, we move with probability $(p_{t,i} - p_{t+1,i})/p_{t,i}$ to some a randomly chosen other expert $j \neq i$ such that $p_{t,j} < p_{t+1,j}$ (choosing j with probability in proportion to $p_{t+1,j} - p_{t,j}$). If we use this switching strategy, then we maintain the probabilities $p_{t,i}$ while paying exactly $d\sum_{i:p_{t,i}>p_{t+1,i}}(p_{t,i} - p_{t+1,i})$ for movement in round t.

Theorem 4 *Say that the on-line algorithm loses d each time it switches experts. If the best expert incurs a total loss of m, then* Hedge *his expected loss at most*

$$\mathbf{E}\left[loss_{\textsf{Hedge}}\right] \leq \frac{1 + d\ln 1/\beta}{1-\beta}\left(\left(\ln\frac{1}{\beta}\right)m + \ln n\right) .$$

Remark. To make better sense of this, let $\beta = 1/(1+\varepsilon/d)$. Then the above bound translates to approximately $(1+\varepsilon)m + (1+1/\varepsilon)d\ln n$. That is, we maintain the same tradeoff as in Theorem 2, with the "adaptation" term multiplied by a factor of d, as one might expect.

Proof. We show that for each loss vector σ_t, the total amount spent on movement is at most $d\ln 1/\beta$ times the loss incurred by the algorithm that step. Combining this fact with Theorem 2, we get the desired bound. Say that before receiving loss vector σ_t we are at expert i with weight $w_{t,i}$ and probability $p_{t,i}$. Let W_t represent the total weight at time t, $\sum_{i=1}^{n} w_{t,i}$.

At each step t we expect to pay $d\sum_{i:p_{t,i}>p_{t+1,i}}(p_{t,i} - p_{t+1,i})$ for movement; the claim is that this is at most $d\ln 1/\beta$ times the task-processing cost at step t, $\sum_i p_{t,i}\sigma_{t,i}$.

$$\begin{aligned}
d\sum_{i:p_{t,i}>p_{t+1,i}}(p_{t,i} - p_{t+1,i}) &= d\sum_{i:p_{t,i}>p_{t+1,i}}\left(\frac{w_{t,i}}{W_t} - \frac{w_{t+1,i}}{W_{t+1}}\right)\\
&\leq d\sum_{i:p_{t,i}>p_{t+1,i}}\left(\frac{w_{t,i}}{W_t} - \frac{w_{t+1,i}}{W_t}\right)\\
&\leq d\sum_i\left(\frac{w_{t,i}}{W_t} - \frac{w_{t+1,i}}{W_t}\right)\\
&= d\sum_i\frac{w_{t,i}}{W}\left(1-\beta^{\sigma_{t,i}}\right)\\
&\leq d\sum_i p_{t,i}\left(\sigma_{t,i}\ln\frac{1}{\beta}\right)
\end{aligned}$$

■

3.3 General-metric MTS algorithms

The problem of how well one can do in task systems against an oblivious adversary is a long-standing question. Prior to the work of Bartal, Blum, Burch, and Tomkins, the best-known bound stood at $O(n)$, despite the conjectured existence of an $O(\log n)$-competitive algorithm [BLS92, IS98]. Bartal *et al* demonstrate an algorithm with a bound of $O(\log^5 n)$ [BBBT97].

The algorithm of Bartal *et al* relies on the Bartal's tree-approximation technique [Bar96, Bar98]. The algorithm is to first probabilistically approximate the metric space for a tree (actually, an $\Omega(\log^2 n)$-hierarchical tree, as defined by Bartal [Bar96]). Then the algorithm runs on the tree using a recursive technique.

Besides the tree approximation, the key element to the Bartal *et al* construction is a uniform-space MTS algorithm (called Odd-Exponent here) with an r-unfair competitive ratio of $1 + (2e\ln n)/r$ applied recursively on the tree. The importance of the unfairness comes from the fact that, if each algorithm running recursively on the next lower level has a ratio of ρ_{sub}, then for each perceived dollar spent on task-processing costs in lower levels, the algorithms are actually spending at most ρ dollars (ignoring the muddying amortization issues). We use $r = \rho_{sub}$. When we receive a task vector t, we multiply all components by ρ_{sub}

before feeding it to Odd-Exponent. So Odd-Exponent's unfairness bound applies with all processing costs being multiplied by ρ_{sub}. This gives us

$$\begin{aligned}
&\mathbf{E}\left[\rho_{sub} \cdot cost_{task}(\sigma, \tau_{A,\sigma}) + cost_{move}(\sigma, \tau_{A,\sigma})\right] \\
&\leq \left(1 + \frac{2e \ln n}{\rho_{sub}}\right)\left(\rho_{sub} \cdot cost_{task}(\sigma, \tau) + \rho_{sub} \cdot cost_{move}(\sigma, \tau)\right) + \zeta \\
&= \left(\rho_{sub} + 2e \ln n\right)\left(cost_{task}(\sigma, \tau) + cost_{move}(\sigma, \tau)\right) + \zeta \ .
\end{aligned}$$

That is, the actual expected cost to Odd-Exponent is at most $\rho_{sub} + 2e \ln n$ times the actual cost to any off-line strategy on the same sequence. In effect, then, the competitive ratio increases by $O(\log n)$ on each level of the tree.

4 Proposed problems

The following are problems that appear tractable and can fit in the thesis. Of course, some of the questions are open and will not appear in the thesis without answers.

4.1 Optional paging

One interesting problem is the problem of "servers with excursions" in the uniform space. This has been asked before, but not with the more stringent goal of unfair competitiveness.

Manasse, McGeoch, and Sleator introduce the *k-server* problem, recognized as another fundamental on-line problem [MMS90]. In this problem, we have k servers, each occupying some point of a metric space. Each time step there is a request to a point of the metric space, and the algorithm must specify some server to move to that point. Our goal is to minimize the total distance moved by all the servers.

An important inspiration behind the k-server problem is paging. Here, the points are in a uniform metric, and each point corresponds to a single page. The k servers lie on the k pages that should lie in the cache. Thus a movement corresponds to a page fault, and we want to minimize the number of page faults.

It is for the uniform-metric k-server problem (ie, paging) that Fiat *et al* introduce the Mark algorithm, achieving a competitive ratio of $2H_k$ [FKL+91]. Fiat *et al* also show that the best-achievable competitive ratio for paging is H_k (which later algorithms achieve [MS91, ACN96]).

In the problem of *servers with excursions* (also proposed by Manasse, McGeoch, and Sleator), each request comes with a cost of δ. The algorithm now gets a choice of serving the requested point (moving a server at a cost according to the distance metric) or refusing to serve it (paying δ). For the uniform-metric unfairness question, the optimal cost is computed with a cost of r to move between points while the on-line algorithm pays only 1. We are looking for an algorithm with r-unfair competitive ratio $O(1 + (\log k)/r)$.

This initially seems like an unnatural question. Reinterpreted, this sort of bound works as a way of providing a bound that is provably better on many inputs than the simple $O(\log k)$ guarantee that Mark provides. We can interpret the $O(1 + (\log k)/r)$ bound by separating the costs. For any request sequence σ, such an algorithm pays at most

$$\mathbf{E}\left[cost_A(\sigma)\right] \leq O(1) cost_{task} + O(\log k) cost_{move} \ ,$$

for any strategy on σ where $task$ is the total cost paid by the strategy for refusing tasks (the δ costs) and $move$ is the total cost paid by the strategy moving servers. Thus, in a sequence that could best be served with lots of renting, our algorithm is constant-competitive, but it is never worse than $O(\log k)$-competitive.

For example, consider a Web search engine that caches responses to queries. There is a small slowly-changing core of frequently-repeated queries that should be represented in the cache, and there are many noisy one-time requests that should never be cached. One can use the $O(1 + (\log k)/r)$ guarantee to get a constant guarantee for this scenario.

In work with Blum and Kalai (currently unpublished), we achieve this guarantee using a variant of Mark employing the multiplicative-update experts algorithm as a subroutine. Unfortunately, our algorithm is neither fast (exponential-time) nor particularly elegant. We hope to either improve this algorithm or to prove a bound on a different, simpler algorithm.

4.2 Multi-armed bandits with switching costs

Auer, Cesa-Bianchi, Freund, and Schapire analyze the *multi-armed bandit problem* [ACBFS98]. This is a limited-information variant of the Experts-DTF problem where at each time step we see only the performance of the expert we chose for that time step. (Think of the experts as slot machines.) The bandit problem fundamentally explores the issue of exploration versus exploitation, balancing the task of discovering which expert is doing well against that of avoiding loss based on what we have seen. Auer *et al* present an algorithm whose expected total loss after T time steps is at most $O(\sqrt{Tn \log n})$ more than the best expert's total loss.

In the spirit of the hybrid Experts-DTF problem where we add a cost to switch between experts (Section 3.2), we can also ask the bandits question for a situation where switching experts has a cost of d. The analysis of Auer *et al* does not apply directly, then, but preliminary work shows that it can be modified to obtain a regret bound of $O(T^{2/3}(dn \log n)^{1/3})$.

Again, this switching-cost addition has implications in competitive analysis, namely for situations where we wish to do nearly as well as the best of several on-line strategies but do not have the resources to simulate all of them at all times. This analogy is not extremely strong, however: Instead of a switching cost, what we have is that each time we switch we may have to do very poorly for a while before we begin to see the real performance of the new expert. For example, if we switch from a LRU paging strategy to LFU, there may be a startup cost associated with LFU while it gathers statistics, during which time it does very poorly but then does much better. Although a switching cost roughly abstracts this situation, it does not address the difficulty of collecting accurate information about a new expert's loss. Whether one can say something about this exact problem remains an interesting question.

5 Summary of goals

The proposed thesis will expand on the results mentioned above. Roughly a year is scheduled to look for additional results that fit within the framework of the thesis outlined in this proposal. Any such results will also appear in the thesis.

5.1 Tentative outline

1. The expert prediction problem
   - 1.1. Problem statement
   - 1.2. Multiplicative-update algorithm
   - 1.3. Partitioning-bound algorithms

2. The metrical task system problem
   - 2.1. Problem statement
   - 2.2. Competitive ratio
   - 2.3. The uniform metric

2.4. Previous work

3. Relating expert prediction and metrical task systems
   3.1. General relationships
   3.2. The addition of switching costs
   3.3. MTS algorithms
   3.4. Experts algorithms

4. The general MTS problem
   4.1. Metric space approximation
   4.2. A $polylog(n)$-competitive algorithm

5. The k-server problem
   5.1. Problem statement and background
   5.2. Optional paging

6. The bandit problem
   6.1. Problem statement and background
   6.2. The addition of switching costs
   6.3. Implications for competitive analysis

5.2 Timeline

begin	item
17 Feb 1999	work on additional questions
1 Jan 2000	begin writing thesis
1 May 2000	defend thesis

References

[ACBFS98] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. *Submitted for publication*, 1998.

[ACN96] D. Achlioptas, M. Chrobak, and J. Noga. Competitive analysis of randomized paging algorithms. In *Proc 4th European Symposium on Algorithms*, pages 419–430. Springer-Verlag, 1996.

[Bar96] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proc IEEE Symposium on Foundations of Computer Science*, pages 183–193, October 1996.

[Bar98] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proc ACM Symposium on Theory of Computing*, pages 161–168, May 1998.

[BB97] A. Blum and C. Burch. On-line learning and the metrical task system problem. In *Proc ACM Workshop on Computational Learning Theory*, pages 45–53, 1997.

[BBBT97] Y. Bartal, A. Blum, C. Burch, and A. Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *Proc ACM Symposium on Theory of Computing*, pages 711–719, 1997.

[BDBK+94] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Widgderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[BKRS92] A. Blum, H. Karloff, Y. Rabani, and M. Saks. A decomposition theorem and lower bounds for randomized server problems. In *Proc IEEE Symposium on Foundations of Computer Science*, pages 197–207, 1992.

[BLS92] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. *J of the ACM*, 39(4):745–763, 1992.

[FKL+91] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator, and N. Young. Competitive paging algorithms. *J of Algorithms*, 12:685–699, 1991.

[FS97] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J Comp Syst Sci*, 55(1):119–139, 1997.

[HW98] M. Herbster and M. Warmuth. Tracking the best expert. *J Machine Learning*, 32(2), August 1998.

[IS98] S. Irani and S. Seiden. Randomized algorithms for metrical task systems. *Theoretical Computer Science*, 194(1–2):163–182, March 1998.

[LW94] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[Mit82] T. Mitchell. Generalisation as search. *Artificial Intelligence*, 18:203–226, 1982.

[MMS90] M. Manasse, L. McGeoch, and D. Sleator. Competitive algorithms for server problems. *J Algorithms*, 11:208–230, 1990.

[MS91] L. McGeoch and D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816–825, 1991.

[Sei98] S. Seiden. Unfair problems and randomized algorithms for metrical task systems. *Information and Computation*, 1998. To appear.

[ST95] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, February 1995.

[Vov95] V. Vovk. A game of prediction with expert advice. In *Proc ACM Workshop on Computational Learning Theory*, pages 371–383, 1995.